Technical Note: The Triad of Context-Driven Learning

A Different Way to Think About Neural Networks

Neural networks are usually described as learning a complicated nonlinear function.

An alternative viewpoint is that many modern learning systems are built from three interacting mechanisms:

1. **Context**
2. **Conditional computation**
3. **Learning**

Each performs a distinct role.

Together they transform a large, difficult learning problem into many smaller, reusable computations.

The Three Components

1. Context — "Where am I?"

The first stage determines the computational state.

Rather than asking directly what the answer should be, the network first determines **where it is in computational state space**.

Examples include:

- ReLU activation patterns.
- Binary context vectors.
- Random projection signatures.
- LSH codes.
- atan2 wedge indices.
- Decision paths.
- Learned routing variables.

The context acts as a coordinate system describing the current situation.

It does **not** compute the answer.

Instead it determines the computational environment in which the answer will be computed.

---

2. Conditional Computation — "What computation should be active here?"

Once the context is known, the network performs context-dependent computation.

Different contexts activate different computational components.

Examples include:

- selecting subsets of parameters,
- activating different neurons,
- routing through experts,
- choosing different local predictors,
- enabling different computational pathways.

This allows one collection of parameters to perform many different computations depending upon context.

Conditional computation is therefore the mechanism that makes parameter sharing possible.

Without conditional computation, every situation would require its own completely independent model.

---

3. Backpropagation — "How should the computation organize itself?"

Backpropagation is often described simply as a method for computing gradients.

Architecturally, it plays a much deeper role.

Backpropagation gradually organizes the shared computational components so that they become useful across many related contexts.

Over training it discovers:

- which computations should be shared,
- which should specialize,
- which context variables matter,
- which interactions improve prediction,
- which computations can be reused.

The result is a highly organized collection of local computations rather than one enormous unfactored function.

How Factorization Emerges

Imagine trying to learn one gigantic lookup table covering every possible input.

The number of possibilities quickly becomes astronomical.

Instead, the network discovers reusable computational pieces.

Contexts determine which pieces are needed.

Backpropagation gradually improves these pieces through repeated exposure to many related contexts.

Learning therefore becomes a process of computational organization rather than simple memorization.

The Role of Local Predictors

Each local predictor should be viewed as a reusable computational module.

It learns how computation should behave within part of the computational state space.

Its responsibilities include:

- identifying informative context variables,
- ignoring irrelevant context,
- combining multiple context signals,
- exploiting statistical regularities,
- producing useful intermediate representations,
- cooperating with predictors in neighbouring layers.

The predictor is learning computation, not context.

The context tells it where that computation is being applied.

Computational State Space

This viewpoint introduces the idea of computational state space.

Instead of viewing inputs simply as points in data space, the network maps them into computational coordinates.

Nearby computational states often require similar computations.

Distant computational states may require entirely different computations.

The quality of the context representation therefore determines how naturally the overall problem can be decomposed into local computations.

Atlas Neural Networks

Atlas neural networks expose this decomposition explicitly.

The designer independently specifies:

- the context generation mechanism,
- the context bandwidth,
- the local predictors.

This makes context an explicit architectural object rather than an implicit consequence of neuron activations.

Because context is explicit it can be measured directly.

Researchers can study quantities such as:

- context entropy,
- bandwidth,
- redundancy,
- utilization,
- stability,
- coverage of the input manifold,
- frequency of context occurrence.

These become architectural properties rather than hidden emergent behaviour.

ReLU Networks

ReLU networks also fit naturally into this framework.

The activation pattern generated by the ReLU units forms the context.

Conditional computation occurs because different activation patterns select different effective linear computations.

Backpropagation organizes the shared weight matrices so that these context-dependent computations become increasingly useful during training.

Although the context is implicit rather than explicitly designed, the same three-way interaction occurs.

Decision Trees

Decision trees represent the opposite end of the spectrum.

The context is accumulated sequentially through a series of logical decisions.

Each leaf corresponds to a local predictor.

Conditional computation consists of selecting the appropriate predictor.

Learning adjusts the tree structure and leaf predictors.

Compared with context-saturated neural architectures, the effective context bandwidth is relatively small because only the information encountered along one path influences the prediction.

Context Bandwidth

Different architectures provide different amounts of contextual information to their predictors.

Low-bandwidth systems:

- simpler,
- easier to interpret,
- more constrained.

High-bandwidth systems:

- provide richer computational coordinates,
- support more complex interactions,
- require more sophisticated organization by the predictors.

Atlas architectures make this bandwidth an explicit design parameter.

This creates a new architectural dimension alongside depth, width, and parameter count.

A New View of Backpropagation

From this perspective, backpropagation is not merely an optimization algorithm.

It is the mechanism that organizes computation over computational state space.

The context provides the coordinates.

Conditional computation determines which computational components participate.

Backpropagation continually reorganizes those shared components into increasingly effective local computations.

The network gradually discovers a factorization of the overall problem that allows enormous computational complexity to emerge from a relatively compact set of shared parameters.

The Central Insight

The three components form an inseparable triad.

- **Context** provides the coordinates in computational state space.
- **Conditional computation** uses those coordinates to select which computations are active.
- **Backpropagation** organizes the shared parameters into a factorized computation over that coordinate system.

None of these components is sufficient by itself.

It is their continual interaction that gives modern neural networks their remarkable ability to learn compact, reusable computational structure from data.